German University in Cairo
Department of Computer Science
Dr. Haythem O. Ismail

Introduction to Artificial Intelligence, Winter Term 2017
Project 1: Help R2-D2 Escape! [1]

Due: October 12^{th}

1. Project Description

The galaxy is in the midst of a civil war. Spies for the Rebel Alliance have stolen plans from the Death Star; a heavily armed space station capable of destroying an entire planet. Unfortunately, the rebels' ship was soon captured by the Death Star's imperial forces guarding the Death Star. Before the rebels were captured, they were able to hide the plans in the memory of a droid called R2-D2. It is R2-D2's mission to try escaping its prison in the Death Star and deliver the stolen plans to the rebels' planet to help destroy the Death Star once and for all.

In this project, you will use search to help R2-D2 escape. The prison is an $m \times n$ grid of cells. Some of the grid cells are occupied by unmovable obstacles. Luckily for R2-D2, there is an inactivated teleportal (a device for instantaneous travel) in one of the grid cells that can take it to the rebel's planet. However, R2-D2 can not use the teleportal to escape unless it is activated. To activate the teleportal, R2-D2 has to push big rocks onto pressure pads lying all around the grid. Each of the rocks and the pressure pads occupies a single cell in the grid. When all the pressure pads have rocks pushed on top of them, R2-D2 can head straight away to the activated teleportal to escape. R2-D2 can only move in the four directions (north, south, east, and west) and push rocks around, but it can not pass through or push the unmovable obstacles or the teleportal.

Using search you should formulate a plan that R2-D2 can follow to escape. An optimal plan is one with as few steps as possible. The following search algorithms will be implemented and each will be used to help R2-D2:

a) Breadth-first search.

b) Depth-first search.

c) Iterative deepening search.

d) Uniform-cost search.

e) Greedy search with at least two heuristics.

f) A* search with at least two *admissible* heuristics.

Each of these algorithms should be tested and compared in terms of the number of search tree nodes expanded.

You may use the programming language of your choice.

[1] This project's theme is based on Star Wars.

Your implementation should have two main functions `GenGrid` and `Search`:

- `GenGrid()` generates a random grid. The dimensions of the grid, the initial location of R2-D2, as well as the locations of obstacles, rocks, pressure pads, and the teleportal are to be randomly generated. You can assume that the number of rocks is always equal to the number of the pressure pads.
- `Search(`*grid*`,` *strategy*`,` *visualize*`)` uses search to try to formulate a winning plan:
  - *grid* is a grid to perform the search on,
  - *strategy* is a symbol indicating the search strategy to be applied:
    - `BF` for breadth-first search,
    - `DF` for depth-first search,
    - `ID` for iterative deepening search,
    - `UC` for uniform cost search,
    - `GR`i for greedy search, with $i \in \{1, 2\}$ distinguishing the two heuristics, and
    - `AS`i for A* search, with $i \in \{1, 2\}$ distinguishing the two heuristics.
  - *visualize* is a Boolean parameter which, when set to `t`, results in your program's side-effecting a visual presentation of the grid as it undergoes the different steps of the discovered solution (if one was discovered).

  The function returns a list of three elements, in the following order: (i) a representation of the sequence of moves to reach the goal (if possible), (ii) the cost of the solution computed, and (iii) the number of nodes chosen for expansion during the search.

2. Groups: You may work in groups of *at most* three.

3. Deliverables

a) Source Code
- You should implement a data type for a search-tree node as presented in class.
- You should implement an abstract data type for a generic search problem, as presented in class.
- You should implement the generic search algorithm presented in class, taking a problem and a search strategy as inputs.
- You should implement a HelpR2-D2 subclass of the generic search problem.
- You should implement all of the search strategies indicated above together with the required heuristics. A trivial heuristic (e.g. $h(n) = 1$) is *not* acceptable. You should make sure that your heuristic function runs in maximum *polynomial* time in the size of the state representation.
- Your program should implement the specifications indicated above.
- Part of the grade will be on how readable your code is. Use explanatory comments whenever possible

b) Project Report, including the following.
- A brief description of your understanding of the problem.
- A discussion of your implementation of the search-tree node ADT.
- A discussion of your implementation of the search problem ADT.
- A discussion of your implementation of the HelpR2-D2 problem.
- A description of the main functions you implemented.

- A discussion of how you implemented the various search algorithms.
- A discussion of the heuristic functions you employed and, in the case of A*, an argument for their admissibility.
- A comparison of the performance of the different algorithms implemented in terms of completeness, optimality, and the number of expanded nodes. You should comment on the differences in the number of expanded nodes between the implemented search algorithms.
- A list of complete and precise instructions on how to run the program and interpret the output.
- Proper citation of any sources you might have consulted in the course of completing the project. *Under no condition*, you may use on-line code as part of your implementation.
- If your program does not run, your report should include a discussion of what you think the problem is and any suggestions you might have for solving it.

4. Important Dates

Source code. On-line submission by October 12th at 23:59 using the following link `https://podio.com/webforms/19301862/1299688` .

Project Report. You should submit a soft copy of the report with the code. A hard-copy should be also be submitted. You have two options:

a) October 12th by 16:00;

b) October 14th by 16:00 provided that an *identical* on-line version is submitted with the code (by October 12th at 23:59).

Brainstorming Session. In tutorials.